Opinnäytetyö (AMK)

Tietotekniikan koulutusohjelma

Sulautetut ohjelmistot

2018

Isto Hietanen

TELEMETRIAOHJELMISTON LAAJENNUKSEN SUUNNITTELU JA TOTEUTUS

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautetut ohjelmistot

2018 | 24

Ohjaaja: TkL Jari-Pekka Paalassalo

Isto Hietanen

TELEMETRIAOHJELMISTON LAAJENNUKSEN SUUNNITTELU JA TOTEUTUS

Tässä opinnäytetyössä suunniteltiin ja toteutettiin ohjelmistolaajennus Lumistarin LDPS-telemetriaohjelmistoon. Laajennus käsittelee telemetriadatan sisältämää sulautettua asynkronisista datavirtaa, jonka sisältönä on NMEA-muodossa olevaa GPS-dataa. Laajennuksen tehtävänä on parsia nämä NMEA-viestit ja lähettää ne edelleen sarjaportin kautta karttaohjelmalle. Tämän lisäksi GPS-informaatio voidaan esittää myös reaaliaikanäytöllä.

Opinnäytetyö on jaettu kahteen osaan. Ensimmäisessä osassa kerrotaan telemetriasta ja käytetyistä teknologioista. Toisessa osassa käsitellään ohjelmistolaajennuksen suunnittelua ja toteutusta.

Työssä perehdyttiin ohjelmistolaajennuksen toteuttamiseen Lumistarin LDPS-ohjelmistoympäristössä. Työn tuloksia voidaan käyttää hyväksi, kun toteutetaan muita sulautettua datavirtoja käsitteleviä ohjelmistolaajennuksia samassa ympäristössä. Laajennuksen toteutustapana on DLL ja ohjelmointikielenä C++.

Opinnäytetyön tuloksena on vaatimusmäärittelyn mukaisesti toimiva ohjelmistolaajennus.

ASIASANAT:

telemetria, IRIG, GPS, NMEA, ohjelmointi

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded Software

2018 | 24

Instructor: Lic. Tech. Jari-Pekka Paalassalo

Isto Hietanen

DESIGNING AND IMPLEMENTING A SOFTWARE PLUGIN FOR A TELEMETRY APPLICATION

The goal of this project was to design and implement a software plugin for Lumistar LDPS telemetry software. The purpose of this plugin is to extract asynchronous datastream from a telemetry stream. The content of this stream is GPS data in NMEA format. The extracted datastream is sent to the map application via a serial port and additionally GPS data can also be viewed in real time display.

This thesis is divided in two parts. The first part focuses in the theory behind telemetry and technologies which are used. The second part focuses on the implementation of the software plugin.

This thesis concentrated on the implementation a software plugin in Lumistar LDPS environment. The results of this project can also be used for implementing additional software plugins in an LDPS environment. The implementation method was DLL and the programming language was C++.

The outcome of the project was a software plugin which is functioning according to the requirement specification.

KEYWORDS:

telemetry, IRIG, GPS, NMEA, programming

SISÄLTÖ

KUVAT

KUVIOT

KOODIESIMERKKI

KÄYTETYT LYHENTEET

DLL	Dynamic Link Libray Dynaamisesti linkitetty kirjasto Ohjelmistolaajennuksen toteutustapa
GPS	Global Positioning System Paikannusjärjestelmä
IRIG	Inter-Range Instrumentation Group Standardi telemetriatiedon välittämiseen
LDPS	Lumistar Data Processing System Lumistarin valmistama telemetriaohjelmisto
NMEA	National Marine Electronics Association Standardi GPS-datan välittämiseen
PCM	Pulse-code modulation Pulssikoodimodulaatio Digitaalisen sarjamuotoisen signaalin koodausmenetelmä

1 JOHDANTO

Telemetria eli kaukomittaus on mittaustietojen välittämistä mittauspaikalta yleensä langattomasti mittausten käsittelypaikkaan. Tyypillisiä telemetrian käyttökohteita ovat mm. tuotekehitys, tutkimus ja testaustoiminta.

Tässä työssä suunnitellaan ja toteutetaan ohjelmistolaajennus Lumistarin LDPS-telemetriaohjelmistoon. Laajennuksen toteutustapana on DLL. Syötteenä oleva telemetriadata sisältää sulautetun asynkronisen datavirran, jonka sisältönä on NMEA-muodossa olevaa GPS-dataa. Tämä datavirta erotetaan ja lähetetään edelleen sarjaportin kautta karttaohjelmalle. Lisäksi NMEA-viesteistä purettu data esitetään myös reaaliaikanäytöllä. Tehtävän toimeksiantajana on Turun ammattikorkeakoulun radiolaboratorio.

LDPS-telemetriaohjelmiston perustoiminnallisuus riittää hyvin tavallisen datan käsittelyyn sekä sen esittämiseen reaaliaikanäytöllä mutta monimutkaisemmissa tapauksissa, kuten sulautettujen datavirtojen käsittelyssä tarvitaan, erillistä ohjelmistolaajennusta datan käsittelyyn ja esittämiseen. Ohjelmistolaajennuksen kautta ohjelmistoon voidaan lisätä sellaisia uusia ominaisuuksia ja toimintoja, joita siinä ei alun perin ole ollut käytettävissä.

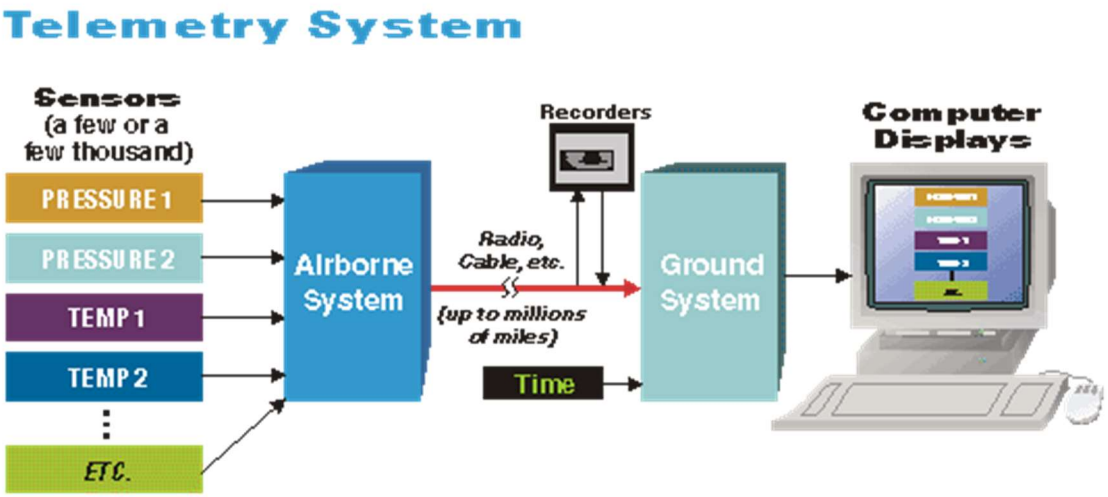
2 TEKNOLOGIAT

2.1 Telemetria

Telemetria eli kaukomittaus on mittaustiedon välittämistä mittausten keräyspaikasta mittausten käsittely- ja seurantapaikkaan. Nimitys on johdettu kreikan kielisistä termeistä tele = kauko ja metron = mittaus. [1]

Telemetriajärjestelmä on tyypillisesti hyvin pitkällä automatisoitu ja mittaustietoa kerätään samanaikaisesti useista, mahdollisesti jopa sadoista antureista. Mittausdata välitetään tavallisesti langattomasti, sillä seurattava oleva kohde on usein joko kaukana tai muuten hankalasti tavoitettavissa. Telemetria ymmärretään yleensä yksisuuntaisena tiedonsiirtona. Telemetriaa käytetään apuna tutkimuksessa, testauksessa ja tuotekehityksessä. [1]

Kuva 1 esittää tyypillisen telemetriajärjestelmän. Mittauspäässä kerätään mittaustietoa useasta eri anturista. Sitten nämä mittaustiedot multipleksataan sarjamuotoon ja lähetetään vastaanottopäähän radiolinkin kautta. Vastaanottopäässä mittaustiedot puretaan jälleen erillisiksi kanaviksi, minkä jälkeen mittaustietoja voidaan joko seurata suoraan reaaliaikanäytöllä tai tallentaa myöhempää käsittelyä varten. [1]

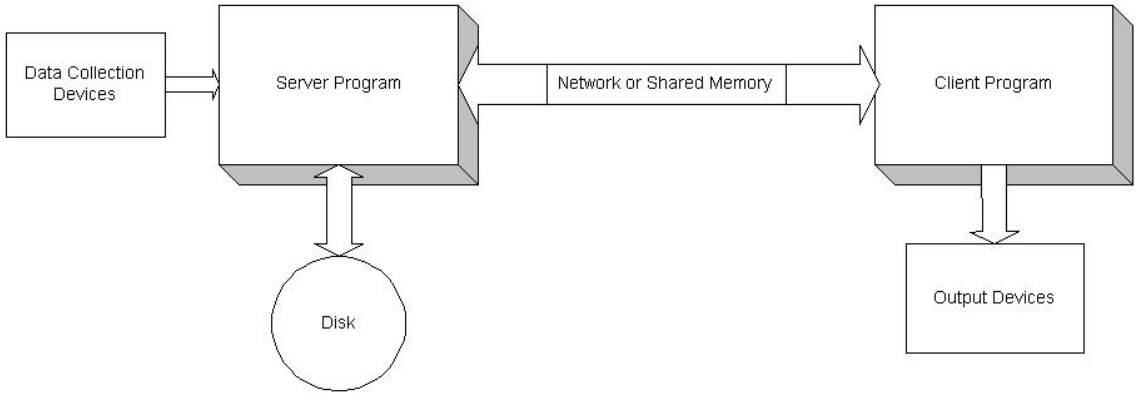


Kuva 1. Tyypillinen telemetriajärjestelmä. [1]

2.2 Telemetriaohjelmisto

Tässä luvussa kerrotaan, mitä osia telemetriaohjelmistoon kuuluu ja mitkä sen perustoiminnot ovat. Esimerkkinä käytetään LDPS-ohjelmistoa. Telemetriaohjelmiston perustoimintoja ovat: mittaustietojen keruu, datan tallentaminen kovalevylle myöhempää käsittelyä varten, datan prosessointi ja tulosten esittäminen reaaliaikanäytöllä. [2]

LDPS-ohjelmisto jakautuu asiakas- ja palvelinosaan. Palvelinosa hoitaa telemetriadatan vastaanoton, purkamisen ja tallentamisen. Asiakasohjelma taas on erikoistunut telemetriadatan käsittelyyn ja esittämiseen reaaliaikanäytöllä. Asiakasohjelma voi olla samassa tietokoneessa kuin serveriohjelma ja tällöin telemetriadata välitys tapahtuu jaetun muistin kautta. Asiakasohjelma voidaan liittää serveriin myös lähiverkon kautta, jolloin useampi asiakasohjelma voi käyttää samaa serveriä. Kuvassa 2 havainnollistetaan LDPS-ohjelmiston asiakas-palvelin rakennetta. [2]



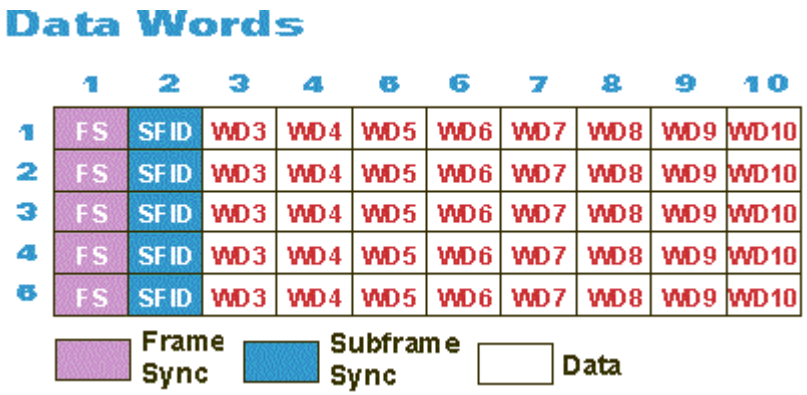
Kuva 2. LDPS-ohjelmiston asiakas-palvelin rakenne. [2]

DLL-laajennusta voidaan käyttää LDPS-ohjelmistossa moneen eri käyttötarkoitukseen. LDPS-ohjelmistossa on valmiita valmistajan tarjoamia DLL-laajennuksia eri tarkoituksiin ja jos näiden tarjoama toiminnallisuus ei riitä, niin käyttäjä voi toteuttaa myös omia laajennuksia. LDPS on tarkoitettu laajennettavaksi, joten siinä on useita valmiita rajapintoja joita voi hyödyntää tekemällä oman DLL-laajennuksen. [3]

2.3 IRIG-106

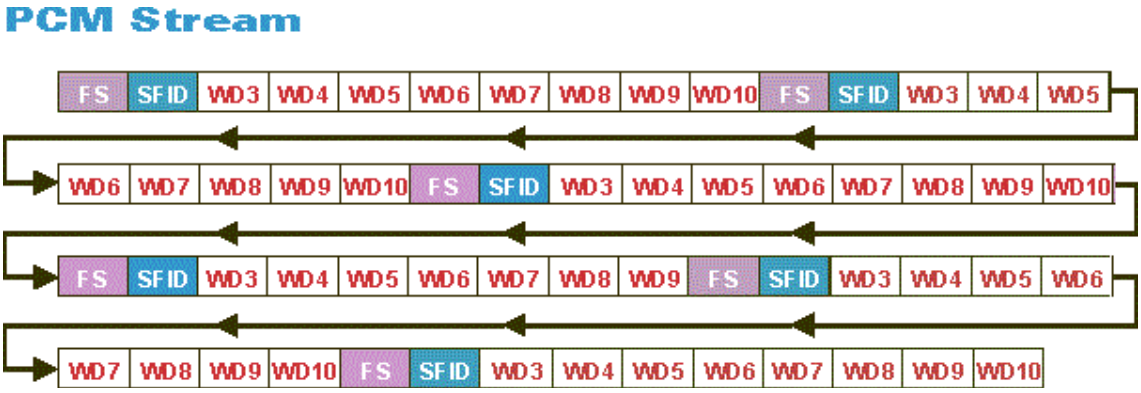
IRIG-106 telemetriastandardi määrittelee laitteisto- ja ohjelmistoriippumattoman menetelmän mittaustietojen lähettämiselle ja vastaanottamiselle. IRIG-106 standardin kappale 4 määrittelee PCM-pulssikoodimoduloidun telemetrian periaatteet. IRIG-106 on menetelmä, jolla monesta eri anturista kerätty mittaustieto voidaan pakata lähetystä ja vastaanottoa varten. IRIG kehyksen muodostavat synkronointimerkit, rivitunnukset ja varsinaiset mittausarvot. Yksittäisellä mittaustiedolla on oma tarkkaan määrätty paikkansa kehyksessä. [4]

IRIG-106 kehyksen rakenne esitetään yleensä taulukko muodossa kuten kuvassa 3. Koko kehystä sanotaan pääkehykseksi (Major frame) ja yksittäistä riviä alikehykseksi (Minor frame). Kullakin rivillä on oma synkronointimerkki (Frame Sync), jolla tunnistetaan rivin alkukohta ja rivitunniste (Subframe identification), jolla identifioidaan yksittäinen rivi.



Kuva 3. IRIG 106-Kehys. [1]

Telemetriadataa lähetettäessä liitetään IRIG-kehyksen rivit yhteen sarjamuotoiseksi bittivirraksi, joka sitten lähetetään eteenpäin pulssikoodimoduloituna. PCM-datavirran muodostamista havainnollistetaan kuvassa 4. Telemetriadatan vastaanotossa on tiedettävä tarkkaan lähetyksessä käytetyt IRIG kehyksen asetukset, jotta telemetriadatan vastaanotto ja purkaminen olisi mahdollista.

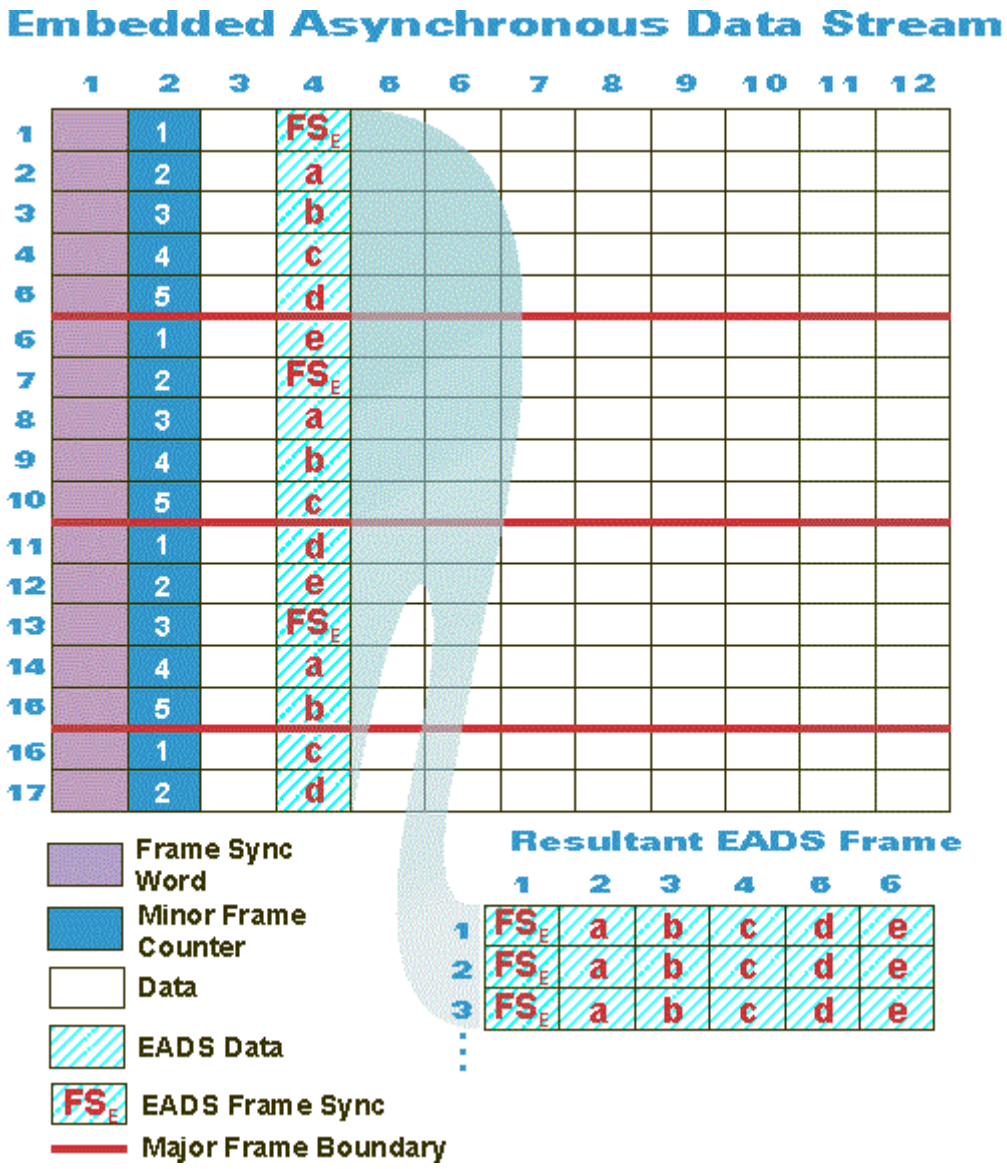


Kuva 4. PCM-datavirta. [1]

2.4 Sulautettu asynkroninen datavirta

Määrätyllä paikalla sijaitsevien yksittäisten mittaustietojen lisäksi telemetriadata voi sisältää myös asynkronista sulautettua dataa, jolle varataan oma tila IRIG-kehyksestä. Asynkronisuus tarkoittaa tässä tapauksessa sitä, että vaikka sulautetulle datavirralle on varattu oma paikka kehyksestä, niin sen sisällölle ei ole ennalta määrättyä alku- ja loppukohtaa. Sulautetun datavirran synkronoinnin ei tarvitse myöskään liittyä mitenkään pääkehyksen synkronointiin. Tämän takia sulautetun datavirran käsittely on monimutkaisempaa ja siihen tarvitaan erillistä prosessointia. Eli datavirtaa käsiteltäessä on ohjelmallisesti etsittävä sulautetun datan alku- ja loppukohdat, jotta datan sisältö voidaan parsia onnistuneesti. Tämän takia asynkronisen datavirran käsittelyyn tarvitaan erillinen ohjelmistolaajennus. Kuvassa 5 havainnollistaan asynkroniselle sulautetun dataviran käsittelyä. Esimerkissä sulautetulle datavirralle on varattuna yksi sarake, eli kussakin kehyksen rivissä on vain yksi solu joka sisältää sulautettua datavirtaa. Jatkokäsittelyä varten nämä solut on yhdistettävä, jotta niiden sisältämä data voidaan purkaa.

Asynkroninen sulautettu datavirta on uudempi ja joustavampi tapa välittää telemetriadataa. Alun perin IRIG-standardin mukaiset telemetrialaitteet oli toteutettu puhtaasti elektroniikalla, jolloin yksittäisen mittaustiedon sijainti kehyksessä oli pakko olla kiinteästi määritelty, jotta mittausdatan vastaanotto ja purkaminen olisi mahdollista. Nykyaikaisempi käyttötapa on, että mittaustietoja lähettävässä päässä on yksi tai useampia sulautettuja prosessorijärjestelmiä jotka keräävät oman mittausdatansa ja lähettävät sen eteenpäin sulautettuna datavirtana. Tällöin eri mittaustilanteissa voidaan sulautetun datavirran sisältöä mukauttaa ja lähettää erilaista tietoa mittauksen eri vaiheissa.



Kuva 5. Sulautettu asynkroninen datavirta. [1]

2.5 NMEA 0183

NMEA 0183 on yleisesti käytetty standardi GPS-datan välityksessä GPS-vastaanottimelta karttaohjelmalle sarjamuodossa. GPS-data on vain pieni osa koko NMEA-standardista, joka sisältää paljon muutakin merielektroniikan väliseen kommunikointiin liittyvää. Karttaohjelmat pystyvät hyödyntämään tyypillisesti vain muutamaa 3–5 NMEA-viestiä. [5]

NMEA-standardin mukaan viestit ovat ASCII-merkeistä koostuvia lauseita joiden maksimipituus on 82 merkkiä. Lauseet alkavat $-merkillä ja päättyvät telanpalautus- ja rivinsiirtomerkkeihin. Lauseiden kentät on erotettu toisistaan pilkulla. [5]

3 OHJELMISTOLAAJENNUKSEN SUUNNITTELU JA TOTEUTUS

Tässä luvussa käydään lävitse ohjelmistoprojektin suunnittelua ja projektin lähtökohtia.

3.1 Suunnittelu

Projektissa tehtävänä on suunnitella ja toteuttaa ohjelmistolaajennus Lumistarin LDPS-ohjelmistoon. Projektin lähtökohtana on asiakkaalla oleva tarve telemetriadatan purkamiseen ja jatkokäsittelyyn. Tehtävänannossa määriteltiin halutut perustoiminnot, jotka, ohjelmistolaajennuksen pitää toteuttaa.

3.1.1 Vaatimusmäärittely

Ohjelmistolaajennuksen perustehtävänä on erottaa telemetriadatan sisältämä asynkroninen sulautettu datavirta jatkokäsittelyä varten. Datavirran sisältönä on NMEA-muodossa olevaa GPS-dataa. Jatkokäsittelyssä erotettu NMEA-muotoinen GPS-data välitetään edelleen sarjaportin kautta karttaohjelmalle. Lisäksi puretut paikka- ja liiketiedot voidaan esittää reaaliaikanäytössä omana näyttökomponenttina.

Tämän tapaisessa projektissa on tyypillistä, että vaatimusmäärittely tarkentuu työn edetessä, koska määrittelyvaiheessa ei vielä tunneta kaikkia uuden laitteen ominaisuuksia ja mahdollisuuksia eikä myöskään voida ennustaa kaikkia mahdollisia vastaantulevia ongelmia.

Projektin aikana tarkentui telemetriadatan lisäkäsittelyn tarve, jota kautta ohjelmaan tuli myös uusia toimintoja. Näitä olivat mm. virheenkäsittelyn tarkentuminen, jotta ohjelma toimisi loogisesti myös poikkeavissa tilanteissa. Tämän jälkeen karttaohjelma toimi vakaammin, kun sijaintipiste ei äkillisesti pomppinut kartta-alueen ulkopuolelle. Lisäksi ohjelmaan lisättiin viimeisen paikan toisto, jos GPS-data katkeaa. Tällä vältettiin karttaohjelmassa näkyvän pisteen katomaminen. Uusina ominaisuuksina ohjelmaan lisättiin vielä datan monitorointi ja asetusten hallinta ja NMEA viestien tallennus tekstimuodossa tiedostoon

Telemetriadataa tutkittaessa tuli yllätyksenä sulautetussa datavirrassa oleva outo dokumentoimaton ilmiö, eli joka kuudes merkki toistetaan. Tämä on otettava huomioon datan parsinnassa, eli toistuvat merkit on poistettava, jotta datan jatkokäsittely olisi mahdollista.

3.1.2 Kehitys- ja ohjelmointiympäristö

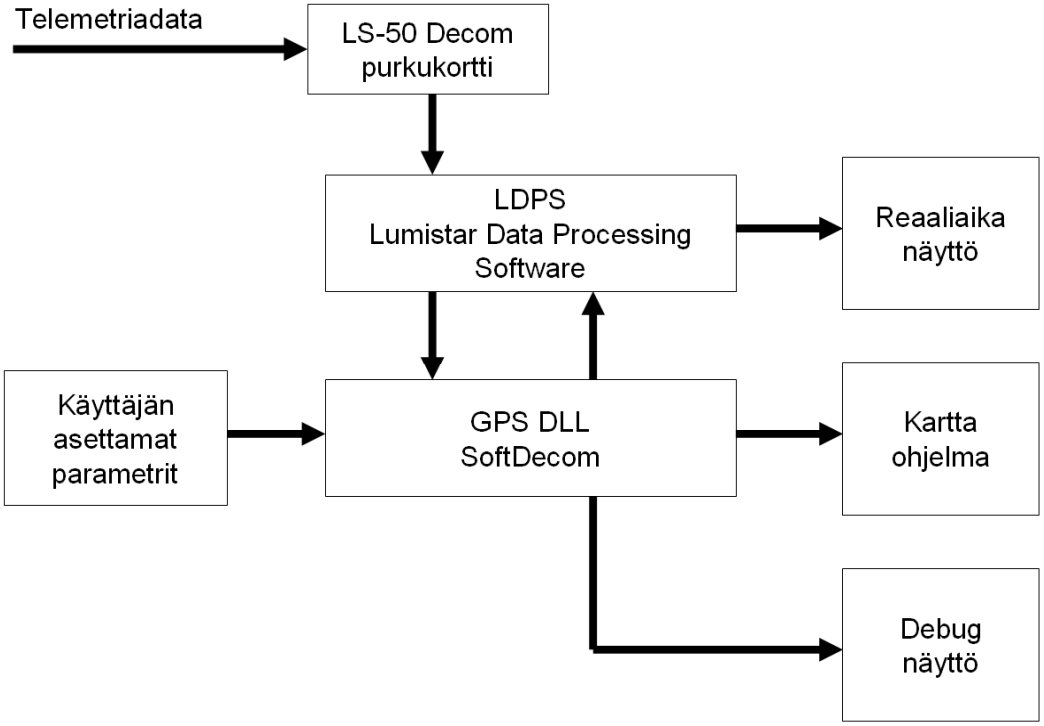
LDPS-ohjelmiston mukana tulevalla CD-levyllä oli esimerkkiprojekteja ja valmiita kirjastoja, jotka on toteutettu Borlandin C++ Builderilla. Tämän takia kehitysympäristöksi valittiin Borlandin C++ Builder 6 ja kehityskieleksi C++. Ohjelmistokehitysympäristönä C++ Builder on jo vanhentunut sekä C++ -version, että koodieditorin kannalta. Käytetty C++ -versio perustuu jo vanhentuneeseen standardiin C++98 ja siinä käytetään myös monia Borlandin omia epästandardeja laajennuksia. Koodieditori on myös varsin kömpelö verrattuna uudempiin kehitysympäristöihin, kuten Visual Studioon. [6]

Telemetriaohjelmisto ja purkukortti toimivat vain Windows XP -ympäristössä, joten kehitys- ja testausympäristönä käytettiin Windows PC -tietokonetta, jossa käyttöjärjestelmäksi oli asennettuna Windows XP. Telemetriadatan käsittelyä varteen tietokoneeseen oli asennettu Lumistarin LS-50 decom -purkukortti ja LDPS Pro Data Processing Software-ohjelmisto. Testauksessa käytettiin DAT-nauhalle tallennettua telemetriadataa, jota voitiin syöttää decom-kortille.

3.1.3 Arkkitehtuuri eli yleiskuvaus

Järjestelmästä tehtiin yleiskuvaus siitä, kuinka ohjelmistolaajennus on liitetty LDPS-ohjelmistoon ja ympäristöönsä. Alun perin tämä kuvaus tehtiin keskustelun lähtökohdaksi, koska sen avulla oli helpompaa kuvailla järjestelmän rakennetta ja toimintaa. Samalla siitä muodostui ensimmäinen kuvaus ohjelmistolaajennuksesta. Kuviossa 1 esitetään ohjelmistolaajennuksen liitäntä ympäristöönsä.

Järjestelmä vastaanottaa telemetriadataa LS-50 decom -purkukortin kautta. LDPS-ohjelmisto vastaanottaa dataa ja välittää sen edelleen ohjelmistolaajennukselle. Sulautetun telemetriadatan käsittely ja purku tapahtuvat kokonaan ohjelmistolaajennuksessa. Ohjelmistolaajennuksen purkamaa dataa voidaan esittää kolmella eri tavalla: reaaliaikanäytöllä, karttaohjelmassa ja debug-näytöllä. Reaaliaikanäytölle menevä GPS-informaatio välitetään takaisin LDPS-ohjelmalle CVT-tietorakenteen kautta. Reaaliaikanäytölle voidaan valita näytettäväksi haluttuja tietoja kuten esimerkiksi kohteen koordinaattitiedot, suunta ja nopeus.



Kuvio 1. Ohjelmistolaajennuksen liityntä ympäristöön.

Ohjelmistolaajennukseen voidaan liittää kaksi sarjaliikenneporttia. Ensimmäinen kytketään karttaohjelmaan ja sen liikenne sisältää pelkästään NMEA-viestejä, joiden tarkistussumma on kunnossa. Tällä tavoin seurattavan kohteen sijainti saadaan näkymään karttanäytöllä. Toinen sarjaportti on tarkoitettu debug-näytöksi ja sen liikenne sisältää koko sulautetun datavirran sisällön, joten siinä näkyvät myös toistuvat merkit sekä NMEA-viestien väleissä olevat täytemerkit. Debug-näyttöä voidaan seurata terminaaliohjelmalla ja sitä tarvitaan ohjelman toiminnan seuraamiseen ja apuna mahdollisten ongelmatilanteiden selvittämisessä.

Ohjelmistolaajennuksen asetusten hallintaa varten on LDPS-ohjelmistossa valmis menetelmä eli käyttäjän asettamat parametrit. Näiden parametrien kautta voidaan asettaa erilaisia alkuasetuksia, jotka halutaan välittää ohjelmistolaajennukselle. Parametrit asetetaan projektin asetusten kautta. Tätä kautta voidaan asettaa mm. sarjaportin numero, valittu dekoodaustapa, yms.

Ohjelmistolaajennus on liitetty LDPS-ohjelmiston asiakasohjelmaan. Tämän asiakasohjelmiston käynnistyessä ladataan projektin asetuksissa määritelty ohjelmistolaajennus ja

suoritetaan alustustoimenpiteet, eli luetaan käyttäjän asettamat parametrit, alustetaan tietorakenteet, käynnistetään säikeet sekä avataan sarjaportit sekä tulostustiedostot. Vastaavasti asiakasohjelmaa suljettaessa kaikki varatut muistialueet vapautetaan, säikeet sammutetaan sekä kaikki avatut portit ja tiedostot suljetaan.

Ohjelmistolaajennuksen ajoitus tulee LDPS-ohjelmiston kautta. Aina kun uutta telemetriadataa on saatavilla, välitetään se edelleen ohjelmistolaajennukselle. Tällöin kullekin toiminnalle on olemassa oma rajattu aikaikkuna, jonka puitteissa kyseinen toiminta täytyy ehtiä suorittaa. Ohjelmoijan on huolehdittava, että oma funktio ei kuluta liikaa aikaa, koska muutoin voidaan hukata dataa.

3.2 Toteutus

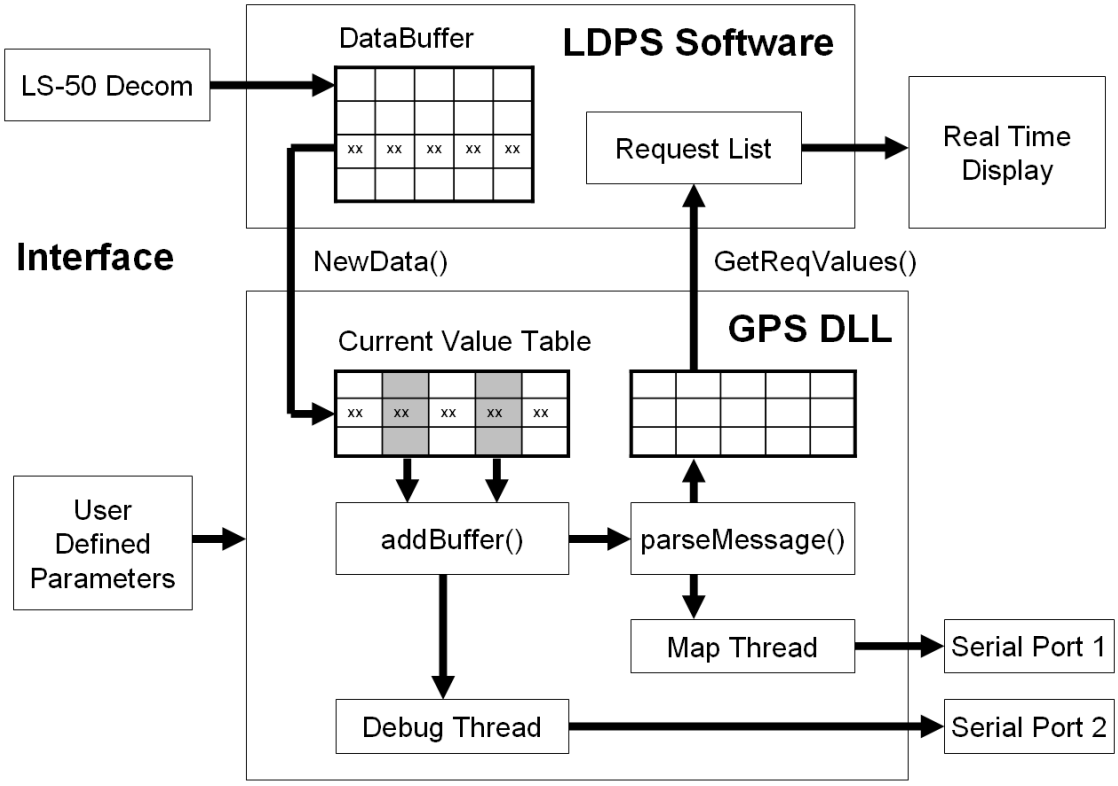
Tässä kappaleessa kuvataan ohjelmistolaajennuksen toteutusta ja sen yksittäisten osien toimintaa tarkemmin. Ohjelmistolaajennus liitetään olemassa olevaan ympäristöön, joten toteutuksessa joudutaan noudattamaan tarkasti tämän ympäristön asettamia vaatimuksia. Oliorakenteesta eli perusluokasta periytetään oma toteutus.

3.2.1 DLL rajapinta

Ohjelmistolaajennuksen rajapinta sisältää 9 funktiota, joista tärkeimmät ovat NewData() ja GetReqList(), joita käytetään telemetriadatan syöttämiseen ja lukemiseen. Ne muodostavat ohjelmistolaajennuksen datankäsittelyn perussyklin. Kutsut toimivat aina pareittain ja ajoitus tapahtuu LDPS-ohjelmiston puolelta. Ensiksi LDPS-ohjelma kutsuu DLL:n funktiota NewData(). Tällöin DLL vastaanottaa yhden uuden rivin (minor frame) telemetriadataa ja käsittelee sen. Seuraavaksi LDPS-ohjelma kutsuu funktiota GetReqList(), jonka kautta luetaan ohjelmistolaajennuksen datankäsittelyn tulokset. Request List –tietorakenne sisältää listan reaaliaikanäytöllä esitettävistä nimetyistä parametreista. LDPS-ohjelma tarkistaa, jos jokin Request Listin sisältämän parametrin arvo on muuttunut, niin tämä päivittynyt arvo välitetään edelleen reaaliaikanäytöllä esitettäväksi. [3]

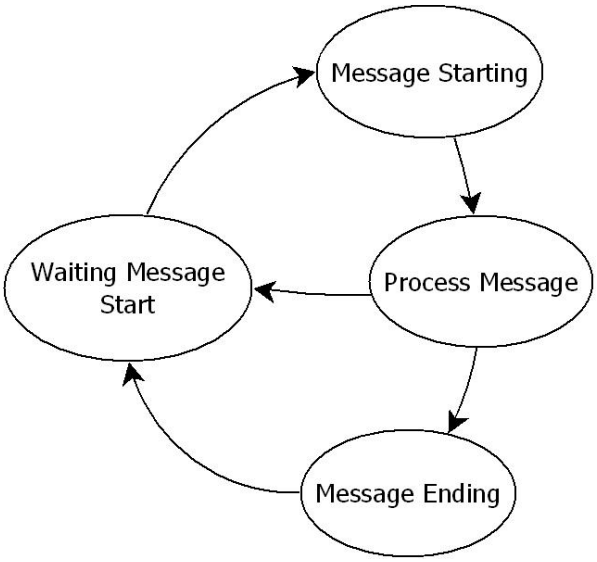
3.2.2 Ohjelmistolaajennuksen toiminta

LDPS-ohjelma kutsuu DDL:n funtiota NewData(). Tällöin DLL saa pointerin puskuriin jonka sisältönä on yksi uusi rivi vastaanotettua telemetriadataa. Tämä datarivi kopioidaan CVT tauluun talteen. GPS-dataa sisältävä sulautettu datavirta muodostuu kahdesta sarakkeesta, joten yksi rivi telemetriadataa sisältää kaksi merkkiä GPS-dataa. Kuviossa 2 kuvataan ohjelmistolaajennuksen toimintaa ja datan kulkua tietovuokaaviona.



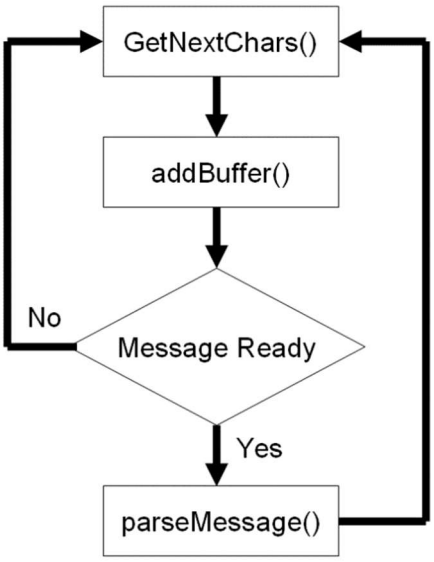
Kuvio 2. Ohjelmistolaajennuksen tietovuokaavio.

AddBuffer()-funktio käsittelee koko sulautetun datavirran. Lisäksi sulautetun datavirran sisältö lähetetään edelleen debug-näytölle. Kullakin NewData()-funktion kutsukerralla funktio saa kaksi uuttaa merkkiä käsiteltäväksi. Kuviossa 3 kuvataan addBuffer()-funktion tilakaaviota. Alkutilana etsitään NMEA-viestin alkua eli $-merkkiä. Kun viestin alkumerkki on löytynyt, aloitetaan viestin käsittely eli lisätään seuraavina saatavat merkit lukupuskuriin, kunnes saavutetaan viestin loppumerkki eli rivinvaihtomerkki.



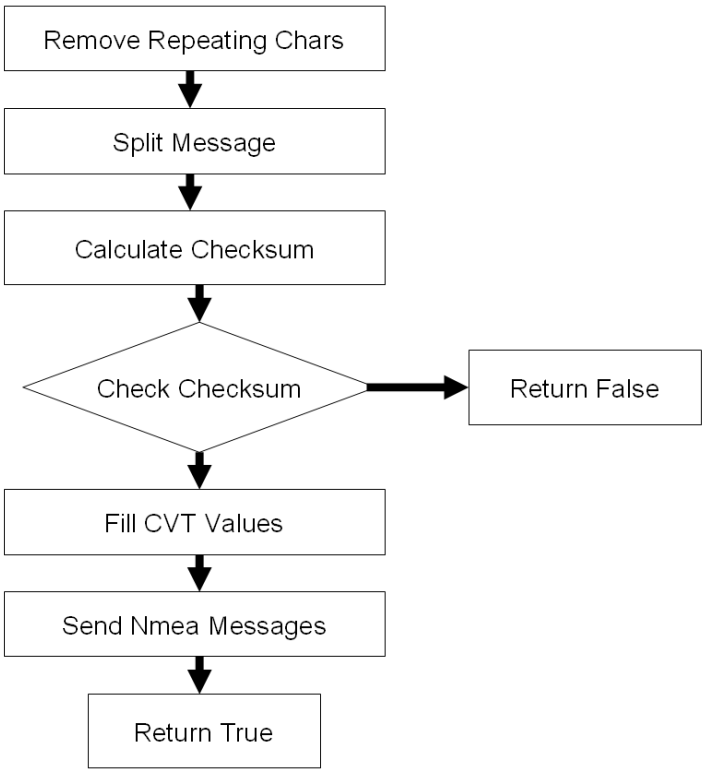
Kuvio 3. addBuffer-funktion tilakaavio.

Tämän jälkeen annetaan lukupuskuriin kerätty raakaviesti parseMessage()-funktiolle jatkokäsittelyä varten. Tämän jälkeen addBuffer()-funktio alkaa jälleen etsiä seuraavan NMEA-viestin alkua. Kuviossa 4 kuvataan NewData()-funktion toimintaa ja addBuffer() ja parseMessage() -funktioiden välistä tiedonkulkua.



Kuvio 4. NewData-funktion vuokaavio.

ParseMessage()-funktiossa käsitellään kerätty NMEA-viesti. Viestissä olevat toistuvat merkit poistetaan. Viesti puretaan erillisiin kenttiin. Viestin tarkistussumma lasketaan ja sitä verrataan viestin mukana tulleeseen tarkisteeseen. Viestit joiden tarkistussumma täsmää välitetään eteenpäin, kun taas muut viestit hylätään. Purettu GPS-data lähetetään CVT-taulun kautta reaaliaikanäytölle ja NMEA-viestit lähetetään sarjaportin kautta karttaohjelmalle. Kuviossa 5 esitellään parseMessage-funktion vuokaavio.



Kuvio 5. parseMessage-funktion vuokaavio.

3.2.3 NMEA-viestin purkaminen

Keskeinen ongelma ohjelmassa oli luotettavan NMEA-viestien parsinta menetelmän kehittäminen. Merkkijonojen parsiminen on monimutkaista ja kaikkien tilanteiden huomioon ottaminen on vaikeata. Valmiista koodista tulee helposti hyvin monimutkaista ja vaikealukuista. Lisäksi virheiden löytäminen monimutkaisesta ratkaisusta on myös hyvin työlästä. Tämän takia tavoitteena oli löytää mahdollisimman yleiskäyttöinen ratkaisu, jossa käytetään mahdollisimman paljon valmiita standardikirjaston toimintoja.

Tämän ongelman ratkaisemiseksi käytettiin valmista koodiesimerkkiä 1. Ensimmäinen funktio palauttaa tuloksen ennalta muodostettuun vektoriin ja toinen funktio palauttaa uuden vektorin.

Valmiin funktion käyttö on mahdollisimman yksinkertaista. Parsinnan suorittavalle funktiolle annetaan NMEA-lause ja funktio palauttaa vector-listan, jossa NMEA-lauseen kentät ovat erillisinä elementteinä.

Koodiesimerkki 1. Merkkijonon jakaminen osiin. [7]

```
#include <string>
#include <sstream>
#include <vector>
#include <iterator>

template<typename Out>
void split(const std::string &s, char delim, Out result) {
    std::stringstream ss(s);
    std::string item;
    while (std::getline(ss, item, delim)) {
        *(result++) = item;
    }
}

std::vector<std::string> split(const std::string &s, char delim) {
    std::vector<std::string> elems;
    split(s, delim, std::back_inserter(elems));
    return elems;
}
```

3.2.4 Poikkeustilanteet

Ohjelmistolaajennusta testattaessa tuli vastaan useita poikkeustilanteita joita ei suunnitteluvaiheessa osattu ennakoida, joten niitä varten jouduttiin tekemään omat erikoisratkaisunsa. Tässä ovat lyhyet kuvaukset havaituista poikkeustilanteista sekä niiden ratkaisutavoista.

Toistuvan merkin ongelma oli ensimmäinen isompi ongelma. Se oli pakko ratkaista, jotta NMEA-viestit saatiin parsittua oikein. Toistuvan merkin poistamista varten pitää ensiksi

tutkia kuinka mones ensimmäinen toistuva merkki on. Datavirran asynkronisesta luonteesta johtuen toistuva merkki voi olla mikä tahansa kuudesta merkistä. Mutta kun poistuvan merkin indeksi on selvitetty, niin ylimääräisten merkkien poistaminen helppoa, kun tarvitsee vain poistaa joka kuudes merkki ensimmäisestä toistuvasta merkistä alkaen. Lisäksi testausvaiheessa huomattiin vielä, että uudempien laitteiden käyttämä enkoodausmenetelmä lisäsi toistuvan merkin joka seitsemännen merkin jälkeen eikä joka kuudennen kuten vanhempi enkoodaus. Tämän takia ohjelmistolaajennukseen lisättiin valinta, että käytetäänkö vanhempaa vai uudempaa enkoodausmenetelmää.

RMC-viestin epäsäännöllisyys aiheutti ongelmia. Testauksessa havaittiin, että karttaohjelma ei tunnistanut RMC-viestejä. Asiaa tarkemmin tutkittaessa havaittiin, että kaikki muut viestit loppuvat standardin mukaisesti telanpalautus ja rivinsiiromerkkeihin, kun taas RMC-viesti loppuu pelkkään telanpalautusmerkkiin. Syynä tähän näytti olevan se, että RMC-viesti ylitti NMEA-viestin maksimipituuden, joka on 82 merkkiä. Tämän takia viestin loppuosa leikkautui pois. Tämä aiheutti pieniä muutoksia ja lisätarkistuksia parsinta-algoritmiin. Käytetty karttaohjelma ei osannut tunnistaa väärin päättyviä viestejä, joten virheellisesti päätyvät viestit piti korjata ohjelmallisesti standardin mukaisiksi.

Viestien välissä olevat tyhjät välit täytettiin täytemerkeillä, jotka voivat periaatteessa olla mitä tahansa merkkejä. Täytemerkkiä toistetaan siihen saakka kunnes seuraava viesti alkaa. Täytemerkiksi näytti valikoituvan satunnaisesti jokin viestissä oleva merkki, eli täytemerkkinä käytettiin eri merkkiä melkein joka välissä. Testausvaiheessa havaittiin, että jos täytemerkkinä oli $-merkki, niin viestien alkukohdan löytäminen oli vaikeampaa. Tämän takia NMEA-viestin alkukohdan etsimiseen piti lisätä tarkistus, joka ohitti toistuvat $-merkit.

Osa ratkaistavaksi tulleista ongelmatilanteista oli luonteeltaan sellaisia, että ne eivät suoranaisesti liittyneet ohjelmistolaajennuksen toimintaan, vaan enemmänkin käyttöympäristöön ja sen toimintatapoihin, esimerkiksi telemetriadatan katketessa ohjelmistolaajennuksen toivottiin toistavan viimeistä sijaintia, koska tällä tavoin karttaohjelmaan saatiin näkymään seurattavan kohteen viimeinen sijainti. Käytetty karttaohjelma hukkasi muutoin viimeisen sijaintitiedon, jollei sitä ei jatkuvasti päivitetty. Ohjelmistolaajennukseen lisättiin toiminto, jolla viimeistä paikkatietoa voidaan tarvittaessa toistaa siinä tapauksessa, että telemetriadatan vastaanotto katkeaa.

Yhtenä kiusallisen ongelmana oli NMEA-datan vastaanotto sarjaportin kautta karttakoneessa. Tietokoneen käynnistyksen yhteydessä Windows-käyttöjärjestelmä saattaa tulkitsee sarjaportin kautta tulevan dataliikenteen sarjahiiren komennoiksi, joka taas saa hiiren osoittimen hyppimään satunnaisesti. Tämä taas vaikeuttaa tietokoneen käyttöä hyvin paljon. Helppona ratkaisuna tähän oli määritellä karttaohjelmalle välitettävän NMEA-liikenteen baudinopeudeksi 9600 bit/s oletuksena olevan 4800 bit/s sijasta, jonka jälkeen Windows-käyttöjärjestelmä ei enää tulkinnut tulevaa dataliikennettä sarjahiireksi.

4 YHTEENVETO

Projekti saatiin vietyä lävitse suunnitellussa laajuudessa. Tuloksena saatiin toteutettua vaatimusmäärittelyn mukaisesti toimiva ohjelmistolaajennus. Aihe oli mielenkiintoinen ja teknisesti haastava. Työssä perehdyttiin ohjelmistolaajennuksen suunnitteluun ja toteuttamiseen Hyötyinä asiakkaalle oli myös lisääntynyt tietämys uudesta LDPS-ohjelmistosta ja sen ominaisuuksista sekä erilaisista käyttömahdollisuuksista. Varsinkin kun sekä telemetriaohjelmisto että käytetty decom-kortti olivat asiakkaalle uusia. Samalla myös telemetriaformaatin yksityiskohtia ja käytännön toteutusta tutkittiin hyvinkin tarkasti.

Aluksi työhön kuului paljon opettelua ja kokeilua, kuinka LDPS-ohjelmisto toimii ja miten sitä käytetään. Miten asetukset tehdään ja kuinka decom-kortti otetaan käyttöön. Tärkeimpinä kohtina ovat, kuinka itsetehty DLL liitetään LDPS-ohjelmistoon ja kuinka telemetriadataa vastaanotetaan ja käsitellään. Sekä LDPS-ohjelmisto että LS-50 -purkukortti olivat uusia myös asiakkaalle, joten työhön kuului paljon itsenäistä tutustumista ja kokeilua.

Ohjelmistolaajennuksen tekeminen alkoi valmiiden esimerkkien tutkimisella ja testiprojektin tekemisellä. Koko projekti oli luonteeltaan iteratiivinen, ja vastaan tulevia ongelmia ratkottiin sitä mukaan, kun niitä ilmeni. Projektin alussa ei ollut selkeätä kuvaa siitä, kuinka paljon aikaa tehtävän suorittamiseen kuluu. Esimerkiksi rajapintaan tutustumiseen ja eri toimintojen kokeilemiseen kului paljon aikaa. Telemetriadatassa oli muutamia dokumentoimattomia ominaisuuksia, jotka aluksi aiheuttivat hyvinkin suurta päänvaivaa, ja sopivien poikkeustilanteiden käsittelyalgoritmien kehittäminen veivät yllättävän paljon aikaa, mutta nämäkin tilanteet ratkaistiin onnistuneesti. Esimerkiksi toistuvat merkit sulautetussa telemetriadatassa sekä virheellisesti päättyvät NMEA-viestit.

Ohjelmalaajennukselta halutut ominaisuudet tarkentuivat matkan varrella useaan kertaan. Myös ohjelman käyttötapaan ja -ympäristöön tuli perehdyttyä varsin kattavasti, joten samalla ratkaistiin monia asiakasta askarruttanutta ongelmaa, Aluksi oli epävarmaa kuinka suuresta työmäärästä olisi kyse mutta lopulta projekti saatiin vietyä onnistuneesti loppuun suunnitellussa laajuudessa. Yhtenä tavoitteena oli tutustua ohjelmaan ja perehtyä kuinka ajurilaajennus toteutetaan tässä ympäristössä. Työn tuloksena saatua ohjelmistolaajennusta voidaan käyttää pohjana toteutettaessa muita vastaavia sulautettuja tietovirtoja käsitteleviä ajureita samassa ympäristössä.

LÄHTEET

[1] L-3 Communications Telemetry West, ”Telemetry Tutorial,” 2012. [Online]. Saatavilla: www.L-3Com.com/TW. [Haettu 25 05 2018].

[2] Lumistar, Inc, ”Technical Manual The LDPS Server Application,” 2014. [Online]. Saatavilla: http://lumi-star.com/uploads/LDPS_UserManual_Part1_Server.pdf. [Haettu 25 05 2018].

[3] Lumistar, Inc, ”Technical Manual The LDPS Client Application,” 2014. [Online]. Saatavilla: http://lumi-star.com/uploads/LDPS_UserManual_Part2_Client.pdf. [Haettu 25 05 2018].

[4] IRIG 106 Community, ”IRIG 106,” 2016. [Online]. Saatavilla: http://www.irig106.org/. [Haettu 25 05 2018].

[5] E. S. Raymond, ”NMEA Revealed,” 2016. [Online]. Saatavilla: http://www.catb.org/gpsd/NMEA.html. [Haettu 25 05 2018].

[6] J. Hollingworth, B. Swart, M. Cashman and P. Gustavson, Borland C++ Builder 6 Developer's Guide (2nd Edition), Sams Publishing, 2002.

[7] S. B. Lippman, J. Lajoie ja B. E. Moo, C++ Primer (4th Edition), Addison-Wesley Professional, 2005.